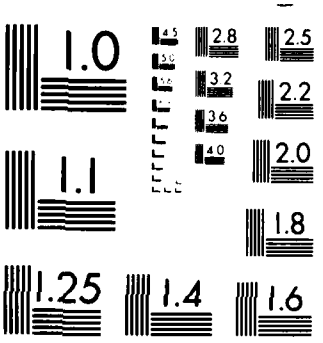
AD-A152 253 PROBABILISTIC ALGORITHMS IN GROUP THEORY(U) HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB J H REIF NOV 84 TR-01-85 N00014-80-C-0647 1/1

UNCLASSIFIED F/G 12/1 NL

END

FILMED

DTIC



(2)

PROBABILISTIC ALGORITHMS IN GROUP THEORY

John H. Reif

TR-01-85

Harvard University

Center for Research in Computing Technology

Aiken Computation Laboratory
33 Oxford Street
Cambridge, Massachusetts 021...

2

PROBABILISTIC ALGORITHMS IN GROUP THEORY

John H. Reif

TR-01-85

PROBABILISTIC ALGORITHMS IN GROUP THEORY

John Reif*

Aiken Computation Laboratory
Division of Applied Sciences
Harvard University
Cambridge, Massachusetts

Laboratory for Computer Science
Massachusetts Institute of
Technology
Cambridge, Massachusetts

November, 1984

*This work was supported by Office of Naval Research Contract N00014-80-C-0647.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Probabilistic Algorithms in Group Theory	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER: TR-01-85
7. AUTHOR(s): John H. Reif	8. CONTRACT OR GRANT NUMBER(s): N00014-80-C-0647
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Harvard University, Cambridge, MA 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, 800 North Quincy Street, Arlington, VA 22217	12. REPORT DATE: November 1984
	13. NUMBER OF PAGES: 30
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Same as above	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

permutation groups, parallel algorithms, probabilistic algorithms, group membership testing, 2-groups

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See reverse side.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

0. ABSTRACT

A finite group G is commonly presented by a set of elements which generate G. We argue that for algorithmic purposes a considerably better presentation for a fixed group G is given by *random generator set* for G: a set of random elements which generate G. We bound the expected number of random elements required to generate a given group G.

Our main results are *probabilistic algorithms* which take as inputs a random generator set of a fixed permutating group $G \leq S_n$. We give $O(n^3 \log n)$ expected time sequential RAM algorithms for testing membership, group inclusion and equality. Our bounds hold for any worse case input groups; we average only over the random generators representing the groups. Our algorithms are two orders of magnitude faster than the best previous algorithms for these group theoretic problems, which required $\Omega(n^5)$ time even if given random generators.

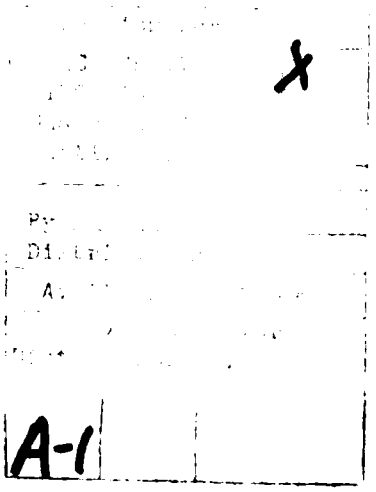
Furthermore, we show that in the case the input group is a 2-group with a random presentation, than those group theoretic problems can be solved by a parallel RAM in $O(\log n)^3$ expected time using $n^{O(1)}$ processors.

0. ABSTRACT

A finite group G is commonly presented by a set of elements which generate G. We argue that for algorithmic purposes a considerably better presentation for a fixed group G is given by *random generator set* for G: a set of random elements which generate G. We bound the expected number of random elements required to generate a given group G.

Our main results are *probabilistic algorithms* which take as inputs a random generator set of a fixed permutating group $G \subseteq S_n$. We give $O(n^3 \log n)$ expected time sequential RAM algorithms for testing membership, group inclusion and equality. Our bounds hold for any worse case input groups; we average only over the random generators representing the groups. Our algorithms are two orders of magnitude faster than the best previous algorithms for these group theoretic problems, which required $\Omega(n^5)$ time even if given random generators.

Furthermore, we show that in the case the input group is a 2-group with a random presentation, than those group theoretic problems can be solved by a parallel RAM in $O(\log n)^3$ expected time using $n^{O(1)}$ processors.



1. INTRODUCTION

1.1 Group Inference and Representation by Random Examples

In informal mathematical discourse, we often make use of examples to illustrate general principles; and in many cases this suffices to convey the essential ideas. For instance, Euclid never formally stated his algorithm for computing GCD but instead explained it by a series of example computations. On the other hand, a student may illustrate comprehension of a general principle, say Euclid's algorithm, by producing some examples.

The fields of Inductive Inference and Combinatorial Enumeration concern the dual problems of inference of a combinatorial structure from examples, and generation of sample examples of a given combinatorial structure, respectively. In Section 2, we investigate these problems when the combinatorial structure of interest is a finite group, and the samples are random, with independent uniform distribution. We give upper bounds on the number of random elements of G required to generate a fixed group (generally, the required number of random samples is a logarithm of the group's order). As an interesting example, consider the group of all permutations of the RUBIC's cube. Our results imply this group can be generated (with high likelihood) by a very small number of random permutations. Furthermore, the results allow us to verify (within high likelihood) the correctness of a "solution method" to RUBIC's cube by applying the "solution method" on a small number random example permutations of RUBIC's cube.

1.2 Group Theoretic Problems

The fundamental groups problems which will concern us are:

(1) *Group Membership:* given an input element x, and a group G, test $x \in G$.

(2) *Group Inclusion:* given groups G, H, test $G \subseteq H$.

(3) *Group Equality:* given groups G, H test $G = H$.

In these problems, a group is normally assumed to be finitely presented. We further assume that a group is randomly presented in the sense defined below.

Let a *random generator set of size* k for group G be a set of k random, independently chosen elements $g_1,\ldots,g_k$ of G, with the condition that $g_1,\ldots,g_k$ generate G. We say $\langle g_1,\ldots,g_k\rangle$ is a *random presentation* of G.

A *probabilistic algorithm* A for a group problem takes as input a random generator sets of given groups. The *expected time complexity* of A is the average time of A over random generator sets, given worse case groups.

Section 2.4 describes a probabilistic algorithm for constructing a strong generator sequence for a randomly presented group. The first use of such probabilistic constructions (in a considerably less general context was due to [Babai, 79].

We also discuss in Sections 2.5 and 2.6 known algorithms for group membership testing, group inclusion, group equality, and random element generation which require strong generator sequences.

1.3 Probabilistic Algorithms for Permutation Groups

The real motivation of our work, and our strongest results, are efficient probabilistic algorithms for the permutation group problems: membership, inclusion, and equality.

[Sim, 78] first gave a well known decision algorithm for these permutation group problems using a construction known as the Sim's Table; others have found it to be very efficient in practice. However, in the worse case, Sim's algorithms were exponential time. [Furst, Hopcroft, and Luke, 81] later modified Sim's algorithm to yield $O(n^6)$ worse case time bounds for these problems. Still further work by [Jerrum, 82] reduced these worse case time bounds to $O(n^5)$. These worse case time bounds seem to be much larger than would be acceptable in practical applications. This situation motivated us to investigate the expected time complexity of permutation group problems.

Our main results are $O(n^3 \log n)$ expected sequential time probabilistic algorithms for permutation group membership, inclusion, and equality. Our time bounds $O(n^3 \log n)$ are only exceeded with probability $n^{-\alpha}$ for some constant $\alpha \geq 1$ that can be set arbitrarily large. In comparison, the previously sighted algorithms for these permutation group problems have expected time complexity which is the same as their worse case complexity (n^5). (Note: Our results here are near optimal in the sense that we can show that further decrease in our sequential time bounds below, say $n^{2.49}$, would imply improvement in the sequential time bounds of best known algorithms for solving a linear system of size $n/2 \times n/2$ over $GF(2)$, which is a special case of permutation 2-group membership testing.)

1.4 Parallel Algorithms in Group Theory

We also investigate the use of parallelism for group theoretic problems. Our goal is efficient parallel algorithms requiring polylog time, polynomial number of processors. In Section 4 we give efficient parallel algorithms for the orbits and block systems of permutation groups. A 2-group is a permutation group whose elements are all of order of a power of 2; they arise naturally since a 2-group is a subgroup of automorphisms of binary trees and furthermore the automorphism group of any trivalent graph with a fixed vertex is a 2-group, see [Luks, 81]. We give in Section 5 efficient parallel algorithms for the problems of membership, inclusion and equality for 2-groups. Our parallel 2-group membership algorithm makes interesting use of our probabilistic techniques: it takes as input a random generator set of a given 2-group, and constructs from them a tower of $O(\log n)$ subgroups with which it is possible to do efficient parallel membership tests.

The parallel complexity of the permutation group membership problem remains open; it is neither known to be log-space complete, for deterministic polynomial time, nor is there a known polylog depth algorithm for group membership. Recently [McKenzie and Cook, 83] have given a polylog time parallel algorithm for Abelian permutation group membership. Their results, and our probabilistic parallel algorithm for 2-group membership, are the only positive results in this regard.

2. PROBABILISTIC ALGORITHMS FOR FINITE GROUPS

2.1 Preliminary Definitions

Let G be group specified by presenting a finite list of generators, as $G = \langle g_1,\ldots,g_k\rangle$. We often assume G has a *finite tower of subgroups* $G_0 \supset G_1 \supset \ldots \supset G_h$ where $G_0 = G$ and $G_h = I$ contains only the identity element. The tower has *height* h. For each $i = 1,\ldots,h$ let $\equiv_i$ be an equivalence relation such that $\forall x,y \in G_{i-1}$, $x \equiv_i y$ iff $y^{-1}x \in G_i$. The blocks of each $\equiv_i$ are the collection of *cosets* of G_i in G_{i-1}, denoted by the quotent G_{i-1}/G_i. Let R_i be a *complete set of coset representations* for G_{i-1}/G_i, ie, a set containing exactly one element from each coset of G_{i-1}/G_i. Since $G = (G_0/G_1)(G_1/G_2)\ldots(G_{h-1}/G_h)$, the sequence of sets $R_1,\ldots,R_h$ is called a *strong sequence of generators* for the group G, with respect to this subgroup tower.

2.2 Elementary Properties of RAND(G)

We will let RAND(G) denote the uniformly distributed random variable giving random elements of G (with equal probability). If x_1,x_2 are random variables, let $x_1 \approx x_2$ if they have the same probability distribution function.

LEMMA 2.1. *If G be a group and $x \in G$, then* $\text{RAND}(G) \approx x \cdot \text{RAND}(G)$.

For proof, observe that the function $f_x(y) = x \cdot y$ is 1-1.

LEMMA 2.2. *Let G' be a subgroup of group G and let R be a complete set of coset representatives for G/G'. Then* $\text{RAND}(G) \approx \text{RAND}(R)^{-1} \cdot \text{RAND}(G')$.

Proof. By Lagrange's Theorem, each coset of G/G' has the same size, so RAND(G) has equal probability of being in any given coset of G/G', say A. By definition $\text{RAND}(R)^{-1}$ also has this property. But producting an element of A by an element of G' keeps us in the same coset A. Hence $\text{RAND}(R)^{-1}\text{RAND}(G')$ has the same

robability of being in A as does RAND(G). By definition, RAND(G) has a niform distribution within each coset A. Let x_A be the unique element of in such that $A = x_A^{-1}G'$. Then since $|A| = |G'|$, we must have AND(A) $\approx x_A^{-1}$RAND(G'). Hence we conclude that RAND(R)$^{-1}\cdot$RAND(G') is also niformly distributed within each coset of G/G'. □

.3 Group Inference from Random Examples

Let G be a fixed finite group. Let $L = (x_1, x_2, \ldots)$ be an infinite ist of elements chosen independently from RAND(G).

HEOREM 2.1. For $0 < \varepsilon < 1$, Prob($G = \langle x_1, \ldots, x_m \rangle$) $\geq 1 - \varepsilon$ if $\geq \log |G| + (\ln(\log |G|))\log(1/\varepsilon_1)$, where $\varepsilon_1 = 1 - (1-\varepsilon)^{1/\log |G|}$.

roof. Let $J = \{j \in \{1,2,\ldots\} : x_j \in \langle x_1, \ldots, x_{j-1} \rangle\}$. List $J = \{j_1, \ldots, j_{|J|}\}$ n increasing order. With probability 1, this gives a tower of subgroups $= G_0 \supset G_1 \supset \ldots \supset G_{|J|-1} \supset G_{|J|} = I$ where $G_{|J|-s} = \langle x_{j_1}, \ldots, x_{j_s} \rangle$. By Lagrange's heorem $j_s \leq |G_{s-1}|/2 \leq |G|/2^s$ and $J \leq \log |G|$. Hence rob$(j_s > j_{s-1} + 1) \leq 1/2^s$ and so Prob$(j_s > j_{s-1} + s) \leq 1/2^{s^2}$ for $s \geq 1$. hus $j_s - j_{s-1} \leq (1/s)\log(1/\varepsilon_1)$ with probability $\geq 1 - \varepsilon_1$. But then with robability $(1-\varepsilon_1)^{|J|} \geq 1 - \varepsilon$ we have $G = \langle x_1, \ldots, x_m \rangle$ if

$$m = |J| + \sum_{s=2}^{|J|} (j_s - j_{s-1} - 1) \leq |J| + \sum_{s=2}^{|J|} \frac{1}{s} \log(1/\varepsilon_1)$$

$$\leq |J| + (\ln |J|)\log(1/\varepsilon_1)$$

$$\leq \log |G| + (\ln(\log |G|))\log(1/\varepsilon_1) \text{ since } |J| \leq \log |G| . \quad \square$$

POLYLOG PARALLEL TIME ALGORITHMS FOR 2-GROUPS

A finite group G is a 2-group if every element has order a power of
We will show that any 2-group has a certain tower of $h = \lfloor \log n \rfloor$ subgroups
$;^{(0)} \supset G^{(1)} \supset \ldots \supset G^{(h)} = I$. Given generators for each subgroup in such a
:r, we can test membership in $O(\log n)^3$ time using $n^{O(1)}$ processors.
:hermore, if G is given by random presentation, we show that we can
struct such a tower in $O(\log n)^3$ time using $n^{O(1)}$ processors.

Parallel Computation of a Structure Forest

Any 2-group G can be decomposed into a subgroup of natural direct
ducts of iterated wreath products. Thus there is a *structure forest* F_G
complete binary trees such that G is a subgroup of the natural direct
duct of the automorphism groups of the trees of F_G. In particular, each
e of F_G is called a *structure tree* and its set of leaves is an orbit of
Furthermore, let ST be any tree or subtree of F_G which is not a leaf.
B be the set of leaves of ST and let B_1, B_2 be the sets of leaves
the two immediate subtrees of ST. Then we require that $\{B_1, B_2\}$ be a
lock system in B.

Suppose we are given generators $g_1, \ldots, g_k$ of 2-group $G \subseteq S_n$. By
cuting first the G-orbit algorithm, and then executing in parallel for
$a, b \in \{1, \ldots, n\}$ the G-block algorithm of Section 4.2, the structure
est F_G can be constructed immediately in $O(\log n)$ further time by
mination of each G-block.

MA 5.1. *The structure forest F_G of 2-group $G \subseteq S_n$ can be constructed in*
not exceeding $O(\log n)$ time using $n^{O(1)}$ processors.

THEOREM 4.2. *A Sir's Table can be constructed from a random presentation of a given permutation group in* S_n, *in expected time* $O(n \log n)$ *using* n^2 *processors.*

COROLLARY 4.1. *For (some easy) permutation groups in* S_n, *given by random presentation, permutation group membership, inclusion and equality can all be done in expected time* $O(n \log n)$ *using* n^2 *processors.*

Suppose G is presented as $\langle g_1,\ldots,g_k\rangle$. For distinct $a, b \in \{1,\ldots,n\}$, et us construct the undirected graph with vertex set $\{1,\ldots,n\}$ and edge t $E_{a,b} = \{\{a,b\}\} \cup \{\{g_i(a),g_i(b)\} \mid 1 \leq i \leq k\}$.

MMA 4.2. [Atkinson, 75] *The connected component of* $(\{1,\ldots,n\},E_{a,b})$ *ntaining* a *is the smallest* G*-block containing* $\{a,b\}$.

Hence finding G-blocks can be efficiently reduced to undirected graph onnectivity, which can apply Lemma 4.1 to get

EMMA 4.3. *The* G*-blocks can be computed (in the worst case) in time* $O(\log n)$ *sing* n^3k *processors.*

emark. If G is given by random presentation, G-blocks can be found in xpected time $O(\log n)$ using $n^2 \log n$ processors.

.3 Limited Parallelism for General Permutation Group Problems

The group membership algorithm of [Sims, 78] was improved by [Furst, opcroft, and Luks, 80] to be polynomial time. However, it appears to be nherently a sequential algorithm; and can not apparently be speeded up to olylog time by parallelization. We do not get much better if we attempt to irectly parallelize our probabilistic algorithms for general permutation roups. (But our low processor bounds may make our parallel algorithms ore practical.) We first observe:

EMMA 4.4. *Given a Sims Table for a permutation group in* S_n, *we can execute membership test:* $x \in G$? *in time* $O(n)$ *using* n *processors.*

Applying Theorem 4.1 to our algorithm of Section 3.3 for Sim's able construction, we get:

4. PARALLEL ALGORITHMS FOR PERMUTATION GROUPS

4.1 The Parallel Machine Model: Known Results

We will assume the concurrent read, concurrent write parallel RAM described in [Shiloach and Viskin, 82].

LEMMA 4.1 [Shiloach and Viskin, 82] and [Viskin and Tarjan, 84]. *Given an undirected graph of* n *vertices and* m *edges, the connected components, a spanning forest, and a preorder of each tree in the forest can all be computed in time* O(log n) *and* n+m *processors.*

4.2 Parallel Computation of Orbits and Blocks of Permutation Groups

Let $G \leq S_n$ be permutation group over $\{1,\ldots,n\}$. It follows immediately from Proposition 3.1, Corollary 3.1, and Lemma 4.1 that

THEOREM 4.1. *We can compute the orbits of* $G = \langle g_1,\ldots,g_k \rangle$ *in time* O(log n) *using in the worst case* O(nk) *processors. Furthermore, if* G *is given by random presentation, we can compute the* G*-orbits in time* O(log n) *using* n log n *processors, with likelihood* $1 - n^{-\alpha}$ *for any sufficiently large constant* $\alpha > 1$.

G acts *transitively* on its orbits. G is *transitive* if G has only one orbit. Suppose G is transitive. A nonempty set $B \subsetneq \{1,\ldots,n\}$ is a G-*block* if $\forall \pi,\pi' \in H$, $\pi(B) = \pi'(B)$ or $\pi(B) \cap \pi'(B) = \emptyset$. If so, then $\{\pi(B) \mid \pi \in G\}$ is a G-*block system*, and group G acts transitively on each of the blocks of the system. If there are no G-blocks of size at least two, then G *acts primitively*. A G-block system is *minimal* if G acts primitively on each block.

COROLLARY 3.2. *For (worse case) permutation groups in* S_n *given by random presentation, permutation group membership, inclusion and equality can all be done in expected time* $O(n^3 \log n)$. *Furthermore, these bounds hold with probability* $1 - n^{-a}$ *for any sufficiently large constant* $a > 1$.

We can easily compute a spanning tree T_i and its preorder in sequential time $O(n \log n)$ by depth first search. It is easily to verify that for each $j \in V_i$, $r_{i,j}$ is an element of G_{i-1} such that $r_{i,j}(i) = j$. Hence $R_i = \{r_{i,j} | j \in V_j\}$ is a complete set of coset representatives for G_{i-1}/G_i, as required. Furthermore $L_i = \{r_{i,\pi(i)}^{-1} \cdot \pi | \pi \in L_{i-1} - F_{i-1}\}$ is a list of $(n-i-1)(\alpha+1)c \log n$ elements of $RAND(G_{i-1})$. Thus $R_1, \ldots, R_n$ are a Sim's Table with probability $\geq 1 - n^{-\alpha}$.

The most costly step of each iteration is [10], which takes time $O(n^2 \log n)$. Since there are n iterations, the total time is $O(n^3 \log n)$. □

3.4 Solution of Permutation Group Problems Utilizing the Sim's Table

The three lemmas below follow immediately from the discussion in Section 2.5.

LEMMA 3.2. *Given the Sim's Table of a permutation group* $G \subseteq S_n$, *and an input* $x \in S_n$, *Sim's membership test:* $x \in G$? *takes sequential time* $O(n^2)$.

LEMMA 3.3. *Given permutation group* $G_1 = \langle g_1, \ldots, g_{k_1} \rangle$ *and the Sim's Table for permutation group* G_2 *where* $G_1, G_2 \subseteq S_n$, *then the group inclusion test:* $G_1 \subseteq G_2$? *takes sequential time* $O(nk_1)$.

Proof. It suffices to test $g_i \in G_2$ for $i = 1, \ldots, k_1$. □

Lemma 3.4. *Given permutation groups* $G_1 = \langle g_1, \ldots, g_{k_1} \rangle$ *and* $G_2 = \langle h_1, \ldots, h_{k_2} \rangle$ *in* S_n *and their Sim's Tables, then we can test group equality:* $G_1 = G_2$? *in sequential time* $O(n(k_1+k_2))$.

As an immediate consequence of the above Lemmas 3.2-3.4, Theorem 2.1, and Corollary 3.1, we have:

begin

Let L_0 be a list of m' random elements independently drawn from RAND(G)

for $i = 1,\ldots,n$ do

begin

[1] Let F_{i-1} be the set consisting of the first $(\alpha+1)c \log n$ elements of L_{i-1}

[2] Compute the connected components of the graph $(\{1,\ldots,n\}, \cup_{\pi\in F_{i-1}} E_\pi)$.

[3] Let V_i be the connected component containing i.

[4] Compute a spanning tree T_i of component V_i rooted at i.

[5] Label each edge $e \in T_i$ with a permutation $\ell(e) = \pi$ where $\pi \in F_{i-1}$ and such that $e \in E_\pi$.

[6] Let $r_{i,i}$ be the identity permutation

[7] In a preorder traversal of tree T_i, compute for each $j \in V_i - \{i\}$ the permutation $r_{i,j} = r_{i,j'} \cdot \ell(j',j)$ where j' is the parent of j.

[8] $R_i \leftarrow \{r_{i,j} \mid j \in V_i\}$.

[9] $L_i \leftarrow \emptyset$.

[10] for each $\pi \in L_{i-1} - F_{i-1}$ do add $r_{i,\pi(i)}^{-1} \cdot \pi$ to L_i.

end

return the Sim's Table $R_1,\ldots,R_n$.

For proof of this algorithm, let us assume inductively for some $i > 1$ that L_{i-1} is a list of $(n-i)(\alpha+1)c \log n$ elements of $\mathrm{RAND}(G_{i-1})$. By Theorem 3.1, V_i is the G_i-orbit containing i, with probability $\geq 1 - n^{-\alpha}$.

Let $G^* = (\cdots(G_{\pi_1})_{\pi_2}\ldots)_{\pi_{\alpha c \log n}}$. $\tau(G^*)$ is just the number of connected components of graph $(\{1,\ldots,n), E_{\pi_1} \cup \ldots \cup E_{\pi_{\alpha c \log n}})\}$ which are proper subsets of orbits of G.

Lemma 3.1 implies for any sufficiently large constant α above c, that $\text{Prob}(\tau(G^*) = 0) \geq 1 - \frac{1}{n^\alpha}$, which proves our Theorem 3.1. □

3.3 Constructing a Sim's Table in $O(n^3 \log n)$ Expected Sequential Time

Fix a permutation group $G \subseteq S_n$ over points $\{1,\ldots,n\}$. For $i = 1,\ldots,n$, let G_i be the subgroup of G fixing points $1,\ldots,i$. The resulting tower $G = G_0 \supset G_1 \supset \ldots \supset G_n = I$ has height $h = n$ and is called the *point stabilizing tower* of G. A strong sequence of generators $R_1,\ldots,R_n$ for this point stabilizing tower is called a *Sim's Table*.

Unfortunately, it is very expensive to construct the Sim's Table by known techniques. [Furst, Hopcroft and Luks, 80] give the first polynomial time algorithm, running in sequential time $O(n^6)$. [Jerrum, 82] improved this time to the best known worse-case bound of $O(n^5)$.

THEOREM 3.2. *Given a random presentation of a given (worse case) permutation group $G \subseteq S_n$, we can construct a Sim's Table in expected sequential time $O(n^3 \log n)$. Further, these bounds hold with probability $\geq 1 - n^{-\alpha}$, for a sufficiently large constant $\alpha > 1$ which can be set arbitrarily large.*

Proof. We will fix $m' = (\alpha+1)cn \log n$, where $c > 1$ is a sufficiently large constant and $\alpha \geq c$ is an arbitrarily large constant.

algorithm of [Hopcroft and Tarjan,73].

COROLLARY 3.1. *We can compute the G-orbits from* $O(\log n)$ *elements of* $RAND(G)$ *in* $O(n \log n)$ *sequential time, with error probability* $< n^{-\alpha}$ *for any sufficiently large constant* $\alpha > 1$.

Now we prove Theorem 3.1. Fix a permutation $\pi_1 \in G$. Let $B_{\pi_1,1},\ldots,B_{\pi_1,k}$ be the orbits of the group $<\pi_1>$. For each $\pi \in G$, let $\varphi_{\pi_1}(\pi)$ be the permutation $\pi' \in S_k$ such that $\forall j = 1,\ldots,k$, $\pi'(j) = j'$ iff $\exists i$ s.t. $\pi(i) \in B_{\pi_1,j}$ and $i \in B_{\pi_1,j'}$. Thus $\varphi_{\pi_1}(\pi)$ is derived from π by collapsing each orbit of π_1 to a single point. Let $G_{\pi_1} = \{\varphi_{\pi_1}(\pi) \mid \pi \in G\}$. Let $\tau(G)$ be the size of the set $\{i \mid \pi(i) \neq i \text{ for some } \pi \in G\}$.

LEMMA 3.1. *If* $\pi_1 \in RAND(G)$, *then* $\text{prob}(\tau(G_{\pi_1}) \leq \tau(G)/2) \geq \frac{1}{2}$.

Proof. Suppose not, then

$$|\{(\pi,i) \mid \pi \in G,\ \pi(i) \neq i\}| < \frac{1}{2}\,|G|\tau(G) \quad .$$

By the pidgeon hole principle, $\exists i_0$ such that $\pi_0(i_0) \neq i_0$ for some $\pi_0 \in G$ but

$$|\{\pi \in G \mid \pi(i_0) = i_0\}| > \frac{|G|}{2} \quad .$$

But this implies that the proper subgroup $\{\pi \in G \mid \pi(i_0) = i_0\}$ of G has order greater than $|G|/2$, a contradiction with Lagrange's Theorem. □

3. PROBABILISTIC ALGORITHMS FOR PERMUTATION GROUPS

Our main results concern permutation group problems. Let S_n denote the group of all permutations over n points.

3.1 Inference of a Permutation Group from Random Examples

Theorem 2.1 implies that for a fixed permutation group $G \subseteq S_n$, $\log(n!) + O(\log n)^2 \leq n \log n - n \log e + O(\log n)^2$ independently chosen permutations from RAND(G) suffice to generate G with probability at least $1-n^{-\alpha}$ for any sufficiently large constant $\alpha > 1$.

3.2 Computing Orbits in Expected Sequential Time $O(n \log n)$

Let $G \subseteq S_n$ be a permutation group over $\{1,\ldots,n\}$. The *G-orbit* of $i \in \{1,\ldots,n\}$ is $\{\pi(i) \mid \pi \in G\}$. Note that the G-orbits partition $\{1,\ldots,n\}$. Let $E_\pi = \{(i,j) \mid \pi(i) = j \text{ or } \pi(j) = i\}$.

It is obvious that if $G = \langle g_1,\ldots,g_k \rangle$ then

PROPOSITION 3.1. *The G-orbits are the connected components of* $(\{1,\ldots,n\}, \bigcup_{i=1,\ldots,k} E_{g_i})$.

We will show:

THEOREM 3.1. *For all* α *above a constant* $c > 0$, *if* $d = \alpha c \log n$ *and if* $\pi_1,\ldots,\pi_d \in$ RAND(G), *then with probability at least* $1 - \frac{1}{n^\alpha}$, *the G-orbits are the connected components of the graph* $(\{1,\ldots,n\}, E_{\pi_1} \cup \ldots \cup E_{\pi_d})$.

We can compute the connected components of a graph of n vertices and $cn \log n$ edges in sequential time $O(n \log n)$ using the depth first search

2.6 Random Element Generation from a Strong Sequence of Generators

Again suppose we have a strong sequence of generators $R_1,\ldots,R_h$ for group G with respect to its subgroup tower $G = G_0 \supset G_1 \supset \ldots \supset G_h = I$. We can compute $\mathrm{RAND}(G)$ by a simple algorithm described in [Hoffman, 82]

begin

for $i = 1,\ldots,h$ let x_i be a random element of R_i

return $x_1 \ldots x_m$

end

To justify this algorithm, observe that Lemma 2.2 implies:

LEMMA 2.5. $\mathrm{RAND}(G) \approx \mathrm{RAND}(R_1)^{-1} \cdot \ldots \cdot \mathrm{RAND}(R_1)^{-1}$.

Remark. It is interesting to observe that a random element of G can be generated by this method in parallel by a binary product tree of depth $O(\log h)$.

Hence we have total failure probability at most $hw(1-1/w)^{m-hw} \leq \varepsilon$. If there is no failure, then each R_i is a complete set of coset representatives for G_{i-1}/G_i. Hence $R_1,\ldots,R_h$ are the strong generators for this subgroup tower. □

2.5 Group Membership Inclusion, and Equality from Strong Generators

Let $G = G_0 \supset G_1 \supset \ldots \supset G_h = I$ be a subgroup tower. Let $R_1,\ldots,R_h$ be a strong sequence of generators of G computed in the previous subsections. We also assume that for any i and $x \in G_{i-1}$ we can effectively find the coset representive $y \in R_i$ such that $x \equiv_i y$ (ie, so $y^{-1}x \in G_i$).

We now describe Sim's algorithm for membership, in the general context of finite groups.

Given an input x and such a strong sequence of generators, the Sim's membership algorithm is:

for $i = 1$ to h do

  if $\exists y \in R_i$ s.t. $x \equiv_i y$ then $x \leftarrow y^{-1}x$

  else return ("x is not a member of G")

return ("x is a member of G").

Remark. Sim's membership algorithm seems inherently sequential, since the parallel time for its execution is at least $\Omega(h)$, where h is the height of the subgroup tower.

Also observe that given another finitely presented group $G' = \langle g_1,\ldots,g_{k'} \rangle$ and a strong generator sequence for G, we can test group inclusion $G' \subseteq G$ by by simply testing $g_i \in G$ for $i = 1,\ldots,k'$. Furthermore, given strong generator sequences for finitely presented groups $G = \langle h_1,\ldots,h_k \rangle$ and $G' = \langle g_1,\ldots,g_{k'} \rangle$, we can test $G = G'$ by varifying $h_i \in G'$ for each $i = 1,\ldots,k$ and $g_j \in G$ for each $j = 1,\ldots,k'$.

```
begin
    Let L_0 be a list of m elements independently drawn from RAND(G).
    for i = 1,...,h do
        begin
            R_i ← ∅;
            for each A ∈ G_{i-1}/G_i do
                begin
                    Let L_{i,A} be the list of elements of L_{i-1} in A
                    if L_{i,A} is not empty than
                        begin
                            choose and delete a random element r_A from L_{i,A}
                            add r_A to R_i
                            for each remaining element x ∈ L_{i,A} do
                                add r_A^{-1} x to L_i
                        end
                end
        end
    return strong sequence of generators R_1,...,R_h
end
```

THEOREM 2.2. *If* $m \geq hw + (\log \varepsilon)/\log(hw(1-1/w))$ *then the algorithm outputs a strong sequence of generators of* G *with error probability* $\leq \varepsilon$.

Proof. We inductively assume that on the i-th iteration, we have had no failure and L_{i-1} is a set of at least $m-(i-1)w$ elements independently chosen from $\mathrm{RAND}(G_{i-1})$. Then Lemma 2.4 implies each element of L_i is independently distributed as $\mathrm{RAND}(G_i)$, and clearly $|L_i| \leq |L_{i-1}| + w \leq m - iw$. By Lemma 2.3, the probability of failure at stage i is at most $w(1-1/w)^{m-hw}$.

2.4 Constructing Strong Generators of a Group from Random Examples

Let G be a finite group with subgroup $G' \subseteq G$. We now wish to discover a complete set of coset representatives of G/G'. Again let L be a random list of $m = |L|$ elements chosen from independent draws from $RAND(G)$. Let E be the event that L contains at least one representative of each coset in G/G'.

LEMMA 2.3. *If* G/G' *has* w *cosets, then* $\text{Prob}(E) \geq 1 - w(1 - 1/w)^m$.

Proof. Consider a given coset, say A, of G/G'. Since a random element of G has equal likelihood to be any coset of G/G', $\text{Prob}(L \wedge A = \cdot) = (1 - 1/w)^m$. Hence $\text{Prob}(\text{not } E) \leq w \text{ Prob}(L \wedge A = \cdot) \leq w(1 - 1/w)^m$. □

Let us assume event E. For each coset $A \in G/G'$, fix r_A to be a random element of $L \wedge A$, and let $R = \{r_A \mid A \in G/G'\}$. For each $x \in G$, let $f_R(x) = r_A^{-1}x$ where $x \in A$.

LEMMA 2.4. $f_R(RAND(G)) \approx RAND(G')$.

Proof. Clearly $f_R(x) \in G'$ for each $x \in G$. By definition, $f_R(RAND(G))$ has a distribution function identical to $r_A^{-1}(RAND(A))$ for randomly chosen $A \in G/G'$. Also since $r_A^{-1}A = G'$ for each $A \in G/G'$, we have $r_A^{-1}(RAND(A)) \approx RAND(G')$. Hence $f_R(RAND(G)) \approx r_A^{-1}\, RAND(A) \approx RAND(G')$. □

We now assume group G has subgroup tower $G = G_0 \supset G_1 \supset \ldots \supset G_h = I$ of height h and width $w = \max_i |G_{i-1}/G_i|$. The following procedure constructs a list of strong generators for G with respect to this subgroup tower.

The structure forest F_G aids us in designing an efficient parallel algorithm for 2-group membership testing.

5.2 Root Actions are Linear

We wish to reduce 2-group membership testing to solving linear [illegible] ver $GF(2)[x]$. We will fix throughout Sections 5.2 and 5.3 the 2·

Let $a_1,\ldots,a_r$ be the roots of structure forest F_G. Let [illegible] *act* on root a_i if a_i is not a leaf and π permutates the two children of a_i and otherwise π *stabilizes* a_i. For any $\pi \in S_n$, let $A(\pi) = (A_1(\pi),\ldots,A_r(\pi))^T$ where $A_j(\pi) = 1$ if π acts on the root a_j and otherwise let $A_j(\pi) = 0$. It is easy to verify:

PROPOSITION 5.1. $\forall \pi_1,\pi_2 \in S_n$, $A(\pi_1 \cdot \pi_2) = A(\pi_2 \cdot \pi_1)$.

Proof. $A(\pi_1 \cdot \pi_2) = A(\pi_1) + A(\pi_2) = A(\pi_2) + A(\pi_1) = A(\pi_2 \cdot \pi_1)$. □

Thus the permutations act commutitively on the roots of the structure forest.

To avoid repeated use of the transpose symbol, we will simply let $x \in \{0,1\}^k$ denote a column vector of booleans. Let M be the $r \times k$ boolean matrix such that $\forall i, 1 \leq i \leq k$, the i-th column of M is $A(g_i)$. Let $A(G) = \{A(\pi) \mid \pi \in G\}$.

LEMMA 5.2. $A(G)$ *is the linear space* $\{Mx \mid x \in \{0,1\}^k\}$ *over* $GF(2)[x]$.

Proof. Suppose $\pi \in G$. Let $\pi = g_{j_1} \ldots g_{j_s}$ be a factorization of π to a product of its generators. For $i = 1,\ldots,k$, let x_i be the residue mod two of $|\{t \mid j_t = i\}|$ (ie, $x_i = 0$ if g_i occurs an even number of times in this factorization of π, and $x_i = 1$ otherwise). Then by Proposition 5.1, $A(\pi) = A(g_1^{x_1} \ldots g_k^{x_k})$. But the j-th element of $A(\pi)$ is the mod two sum Σx_i where the sum is taken over just those i such that g_i acts on a_j. Hence by definition, $Mx = A(\pi)$. □

Let $G^{(1)}$ be the subgroup of G consisting of the permutations that fix the roots $a_1,\ldots,a_r$, ie, $G^{(1)} = \{\pi \in G \mid A(\pi) = (0,\ldots,0)^T\}$.

LEMMA 5.3. $\pi \in G$ *iff* $\exists x \in \{0,1\}^k$ s.t. $Mx = A(\pi)$ *and* $(g_1^{x_1} \ldots g_k^{x_k})^{-1}\pi \in G^{(1)}$.

Proof. If $Mx = A(\pi)$ and $(g_1^{x_1} \ldots g_k^{x_k})^{-1}\pi \in G^{(1)}$ then since $g_1^{x_1} \ldots g_k^{x_k} \in G$, we have $\pi \in G$. Suppose on the other hand that $\pi \in G$. By Lemma 5.2, $\exists x \in \{0,1\}^k$ s.t. $Mx = A(\pi)$ and so $A(g_1^{x_1} \ldots g_k^{x_k}) = A(\pi)$, hence $A((g_1^{x_1} \ldots g_k^{x_k})^{-1}\pi) = (0,\ldots,0)^T$ so $(g_1^{x_1} \ldots g_k^{x_k})^{-1}\pi \in G^{(1)}$. □

5.3 2-Group Membership Testing Given a Block Structure Tower

Let the *block-structure tower* of G be the sequence of subgroups $G = G^{(0)} \supset G^{(1)} \supset \ldots \supset G^{(h)} = I$ where $G^{(i)}$ is the subgroup of G containing only permutations that fix all the nodes of depth $< i$ in the structure forest F_G. Since the depth of F_G is at most $\lfloor \log n \rfloor$, this tower has depth $h \leq \lfloor \log n \rfloor$.

THEOREM 5.1. *Suppose we have generators for the block structure tower* $G = G^{(0)} \supset G^{(1)} \supset \ldots \supset G^{(h)} = I$ *of the 2-group* $G \subseteq S_n$. *Then we can test membership in* G *in time* $O(\log n)^3$ *using* $n^{O(1)}$ *processors.*

Proof. Suppose we are input some permutation $\pi \in S_n$. We first must determine the existence of a solution $x \in \{0,1\}^k$ of the linear equation $Mx = A(\pi)$ defined in Lemma 5.2. If no solution exists, we reject π. Otherwise we use this solution x to reduce the problem to membership testing in $G^{(1)}$. In particular, by Lemma 5.3, we need only test if $(g_1^{x_1} \ldots g_k^{x_k})^{-1}\pi \in G^{(1)}$. This is done by recursive application of the membership test. Since $h \leq \lfloor \log n \rfloor$, at most $\log n$ stages suffice. Each stage thus requires solution of a linear system over $GF(2)[x]$ of size at most $n \times n$, which can be done in time $O(\log n)^2$ using $n^{O(1)}$ by the parallel algorithm of [Borodin, von zur Gathen, and Hopcroft,

82]. Thus the total time is $O(\log n)^3$ using $n^{O(1)}$ processors. □

Remark. These time bounds can be decreased to $O(\log n)^2$ if the required matrix inverses are precomputed, (so that each stage requires only matrix multiplications, which takes only $O(\log n)$ time using $n^{O(1)}$ processors).

5.4 Polylog Time Construction of the Block-Structure Tower from a Random Presentation

Let $L = \{\tau_1,\ldots,\tau_{m_o}\}$ be a list of m_o permutations independently chosen from RAND(G). Fix $m = m_o/\log n$. Let Y be the linear space over $GF(2)[x]$ generated by $A(\tau_1),\ldots,A(\tau_m)$. Since $A(G)$ is a group of size at most $2^r \leq 2^n$ by Theorem 2.1 we have

LEMMA 5.4. $\text{Prob}(Y = A(G)) \geq 1 - \varepsilon$ *if* $m \geq r - (\ln(r))\log(1/\varepsilon_1)$, *where*

Note that since the total number of nodes of the structure fore $2n$, we can bound the probability of error to be at most ε using $m = 2n + (\ln(2n))\log(1/\varepsilon_1)$ random elements of G. For example, if $\varepsilon = n^{-k}$, then it suffices to let $m = 2n + o(n)$.

Let M' be the $r \times m$ boolean matrix whose i-th column is $A(\tau_i)$ for $i = 1,\ldots,m$. Then by construction $Y = \{M'x \mid x \in \{0,1\}^m\}$. Let $Y^{(1)},\ldots,Y^{(\ell)}$ be a basis for Y. For $i = 1,\ldots,\ell$, we find $x^{(i)}$ such that $M'x^{(i)} = Y^{(i)}$ and define the permutation $\sigma_i = \tau_1^{x_1^{(i)}} \ldots \tau_m^{x_m^{(i)}}$. Since $A(\tau_i) = Y^{(i)}$, we have by construction $Y = A(\langle\sigma_1,\ldots,\sigma_\ell\rangle)$.

Now let M'' be the $r \times \ell$ boolean matrix whose i-th column is $A(\tau_i)$ for $i = 1,\ldots,\ell$. Again by construction, $Y = \{M''z \mid z \in \{0,1\}^\ell\}$.

For purposes of membership testing, it suffices to have the list $\sigma_1,\ldots,\sigma_\ell$ which generate the coset representatives of $G/G^{(1)}$.

LEMMA 5.5. *The set* $R = \{\sigma_1^{z_1} \ldots \sigma_\ell^{z_\ell} \mid z_1,\ldots,z_\ell \in \{0,1\}\}$ *is a complete set of coset representatives for* $G/G^{(1)}$.

For each $\pi \in S_n$, let $f_R(\pi) = (\sigma_1^{z_1} \ldots \sigma_\ell^{z_\ell})^{-1}\pi$ where $M''z = A(\pi)$. Clearly, if $\pi \in G$ then $f_R(\pi) \in G^{(1)}$. Hence by Lemma 2.1 we have

LEMMA 5.6. $f_R(\text{RAND}(G)) \approx \text{RAND}(G^{(1)})$.

Since $L = (\pi_1, \ldots, \pi_{m_o})$ is a list of independently chosen elements of RAND(G), and we have only utilized the first m elements of L to construct a random presentation of $G^{(1)}$, it follows from Lemmas 2.2 and 5.6 that

LEMMA 5.7. $L^{(1)} = (f_R(\pi_{m+1}), \ldots, f_R(\pi_{m_o}))$ *is a list of independently chosen elements of* $\text{RAND}(G^{(1)})$.

The above Lemma implies that we can repeat the above construction, to construct a random presentation of $G^{(2)}$ from the first m elements of $L^{(1)}$. A further $\log n$ stages yields generators for the entire block-structure tower $G = G^{(0)} \supset G^{(1)} \supset \ldots \supset G^{(h)}$ as required. The linear algebraic computations (such as computing basis vectors), required in each stage of the above construction of $G^{(i)}$, can be done in time $O(\log n)^2$ using $n^{O(1)}$ processors by the methods of [Borodin, van zur Gothen, and Hopcroft, 82]. Since there are at most $\log n$ stages, we have:

THEOREM 5.2. *Given a random presentation of 2-group* $G \subseteq S_n$, *we can construct generators for each subgroup of the block-structure tower, in time* $O(\log n)^3$ *using* $n^{O(1)}$ *processors.*

By Theorems 5.1 and 5.2, we have:

COROLLARY 5.1. *Given a random presentation of (worst case) 2-group* $G \subseteq S_n$, *and some* $x \in S_n$ *we can test membership in* G *in expected time* $O(\log n)^3$ *using* $n^{O(1)}$ *processors.*

COROLLARY 5.2. *Given random presentations of (worse case) 2-group* $G_1, G_2 \subseteq S_n$, *we can test* $G_1 \subseteq G_2$ *in expected time* $O(\log n)^3$ *using* $n^{O(1)}$ *processors.*

References

Atkinson, M.D., "An algorithm for finding the blocks of a permutation group," *Math. of Comp.* 29 (1975), 911-913.

Babai, L., "Monte Carlo algorithms in graph isomorphism testing," *SIAM J. on Computing* (1979).

Borodin, A., von zur Gothen, and J. Hopcroft, "Fast parallel matrix and gcd computations," *Proc. 23rd Annual Symp. FOCS*, Chicago (1982), 65-71. To appear in Information and Control.

Cook, S.A., "Towards a complexity theory of synchronous parallel computation," Presented at *Internationales Symposium über Logik und Algorithmik zu Ehren von Professor Hort Specker*, Zürich, Switzerland, February 1980.

Furst, M., J. Hopcroft, and E. Luks, "Polynomial-time algorithms for permutation groups," *Proc. 21st IEEE Symp. on Foundations of Computer Science*, (1981), 36-41.

Galil, Z., C.M. Hoffmann, E.M. Luks, C.P. Schnorr, and A. Weber, "An $O(n^3 \log n)$ deterministic and an $O(n^3)$ probabilistic isomorphism test for trivalent graphs," *23rd Annual IEEE Symp. on Foundations of Computer Science*, Chicago, Ill. (Nov. 1982), 118-125.

Hoffmann, C.M., "Group-theoretic algorithms and graph isomorphism," Lecture Notes in *Computer Science*, Springer Verlag, New York, (1982).

Hopcroft, J.E. and R.E. Tarjan, "Efficient algorithms for graph manipulations," *Comm. ACM* 16, 6, 372-378 (1973).

Jerrum, M., "A compact representation for permutation groups," *23rd Annual IEEE Symp. on Foundations of Computer Science*, Chicago, Ill. (Nov. 1982), 126-133.

Luks, E.M., "Isomorphism of graphs with bounded valence can be tested in polynomial time," *Proc. 21st Symp. on Foundations of Computer Science*, (1981), 42-49.

McKenzie, P. and S. Cook, "Parallel complexity of the Abelian permutation group membership problem," *24th Symp. on Foundations of Computer Science*, (1973), 154-161.

Pippenger, N., "On simultaneous resource bounds," in *Proc. of the 20th IEEE Symp. on Foundations of Computer Science* (1979).

Sims, C.C., "Computational methods in the study of permutation groups," in *Computational Problems in Abstract Algebra*, J. Leech (ed.), Pergamon Press (1970).

Sims, C.C., "Some group-theoretic algorithms," *Lecture Notes in Math.* 697, Springer-Verlag, Berlin (1978), 108-124.

Shiloach, Y. and U. Vishkin, "An O(log n) parallel connectivity algorithm," *J. Algorithms* 3, 57-67 (1982).

Vishkin, U. and E. Tarjan, "An efficient parallel biconnectivity algorithm," *25th Symp. on Foundations of Computer Science*, Palm Beach, Florida, (1984).

Wielandt, H., *Finite Permutation Groups*, Academic Press, New York, 1964.

END

FILMED

5-85

DTIC